SPFCN: Select and Prune the Fully Convolutional Networks for Real-time Parking Slot Detection

Zhuoping Yu[1], Zhong Gao[1], Hansheng Chen[1], Yuyao Huang[1,2,*]

***Abstract*—For vehicles equipped with the automatic parking system, the accuracy and speed of the parking slot detection are crucial. But the high accuracy is obtained at the price of low speed or expensive computation equipment, which are sensitive for many car manufacturers. In this paper, we proposed a detector using CNN(convolutional neural networks) for faster speed and smaller model size while keeps accuracy. To achieve the optimal balance, we developed a strategy to select the best receptive fields and prune the redundant channels automatically after each training epoch. The proposed model is capable of jointly detecting corners and line features of parking slots while running efficiently in real time on average processors. The model has a frame rate of about 30 FPS on a 2.3 GHz CPU core, yielding parking slot corner localization error of 1.51±2.14 cm (std. err.) and slot detection accuracy of 98%, generally satisfying the requirements in both speed and accuracy on on-board mobile terminals.**

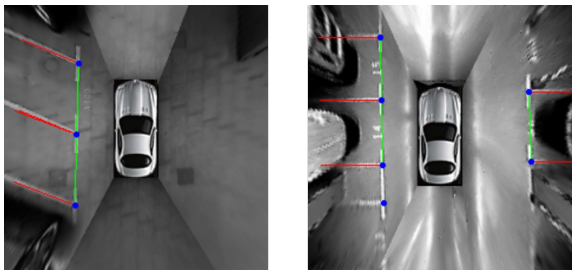
I. INTRODUCTION

With the development of autonomous driving technologies, automatic parking assist system has become an intensive research topic and its first step is to find the parking slot. The proposed solutions can be roughly divided into two categories: one is infrastructure based approach which typically use pre-built maps and sensors, and the other is on-board sensor based approach which use only sensors mounted on vehicles to detect available parking slots. In this paper, we used second approach to detect the parking slot makings which are group of line segments drawn on the ground to indicate an effective parking area.

The detection methods based on locating visual markers can again be divided into two categories: line-based [3] and corner-based [4]. In this paper, a pure visual method is designed to find the parking slots, in which both line features and corner features will be detected to improve accuracy. Our method differs from other parking slot detectors in three major contributions as concluded below:

- First, we introduce the hourglass structure [5] into parking slot detection and discuss and verify the effect of the network with/without stacked hourglass and intermediate supervisions.
- Second, we design a Select-and-Prune Module (SP Module), which can select the most suitable receptive field for convolution, and prune redundant channels automatically. The module can be easily used in training epoch and will not bring extra calculations to the inference stage.
- Third, we use both corner features and line features to jointly infer the parking slot. The joint features bring better stability in inferencing and can maintain a certain accuracy when continuously compressing the model parameters.

[1] Institute of Intelligent Vehicles, School of Automotive Studies, Tongji University, Shanghai, China
[2] State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, China
[*] Corresponding author. E-mail: huangyuyao@tongji.edu.cn
The model will be posted on https://github.com/tjiiv-cprg/SPFCN-ParkingSlotDetection



(a) non-vertical (b) multi-slots

Fig. 1. Detection results with corners marked as blue points, entry lines marked green and separating lines marked red. The synthetic overhead view is composed by images from four calibrated, undistorted and homographic transformed fish-eye cameras mounted on the vehicle, according to [1]. The artifacts and misalignments aused by calibration error can be further reduced by additional transformation [2] and stitching algorithms, but we only focus on the detection pipeline in this paper.

II. RELATED WORKS

Parking Slot Detection: One of the first works to use traditional machine learning method for parking slot detection was presented by Xu et al. [6]. Later implementations of traditional methods include using Radon transform [7] to detect slot lines or using Harris detector [8], AdaBoost [1] to detect slot corners. The results of these methods largely depend on the feature design of classifiers and subsequent logical inference, and they are also sensitive to the size or direction of input images. The occluded and shadowed ground markings also bring extra difficulties in solving the detection problem with traditional methods.

As for the deep-learning based methods, for example, Zhang et al. [4] use YOLO v2 [9] to locate parking slot corners. However, YOLO is a object detection network designed not exclusively for parking slot and running YOLO in real time on CPU can be a serious problem due to its massive number of FLOPs. Today's production cars are often fitted with only CPUs or even MCUs. Hence, there still lies potential in the performance of the backbone itself.

Hourglass-like Network Structure: In this paper, the hourglass structure [5] is selected as our backbone. This

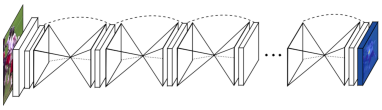


Fig. 2. Stacked hourglass network [5].

architecture is specifically designed for key-point detection and has been widely used in many related tasks such as the human joint detection.

The architecture is shown in Fig. 2. It has the flexibility in stacking the hourglasses to adjust the number of parameters and the speed of network. The input image is first down-sampled, the possible regions of corner-points and lines are found on the smaller feature map, and the feature size is restored by up-sampling, where key areas detected by the previous layer are superimposed on the features of skip, thereby it can increase the response of key areas on the final heat map.

Efficient Convolution Module Design: Regarding basic convolution modules, there is an increasing need for the efficient design. SqueezeNet [10] implemented AlexNet-level accuracy on ImageNet with a 50-fold reduction in parameters. The MobileNet [11] and ShuffleNet [12] use depth-wise separable convolution to compress parameters and FLOPs of the network while ensuring accuracy. Based on ResNet [13] and DenseNet [14], PeleeNet [15] uses a two-way dense layer and a stem block to optimize its performance.

Li et al. [16] realized that the convolution kernel size of visual neurons is rarely considered when constructing the CNN. They propose a dynamic selection mechanism to adaptively adjust the weights of kernels in different sizes when fusing multiple branches containing information of different spatial scales.

Inspired by that, we design a new convolution module which can adaptively select the convolution kernel size and prune the redundant channels. We call it the SP Module and substitute it for the residual modules [13] in the original stacked hourglass network.

III. METHOD

We developed an efficient and practical solution for simultaneous detection of corners and marking lines of parking slots. The general idea is using FCN (Fully Convolutional Networks) to accomplish all the jobs, without the help of FC (Fully Connected) layers. The classification, regression, detection, and segmentation tasks generally can be modeled as a softmax heatmap generator followed by a certain type of proposal (e.g. Region Proposal Network [17], etc.), pooling (e.g. ROI Pooling [17], Line Pooling [18], Average Pooling), and finally thresholding. We then develop a scalable and tailorable training strategy for pre-training the model, selecting from the candidate kernels while training, and pruning selected kernels' channels while fine-tuning. The trained networks with simple post-processing logic can show good efficiency and flexibility to be generalized to other problems by following the similar solution paradigm, the solution for parking slot detection in this paper is an example of such idea.

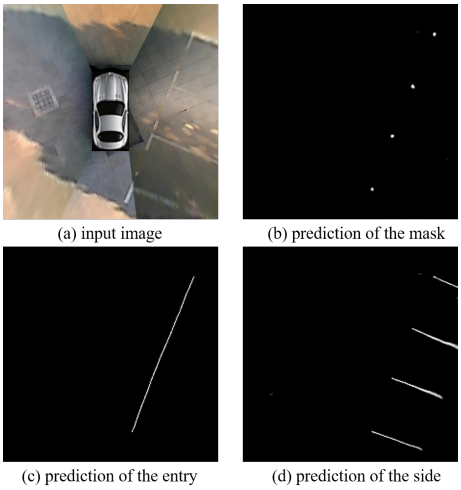


Fig. 3. Predicted heatmaps of the Heads.

A. Network Structure

Among all the FCNs, we choose the Stacked Hourglass Network [5] as the basic architecture, for its scalability to different layers of stacks. The prediction heads of the parking slot detection task produces heatmaps of slots' corners, entrance lines and separating lines, as shown in Fig. 3. The final layers, as well as the intermediate supervision layers, supervise the layers by softmax losses through those heads. The existence of the intermediate supervision layers can assist the back propagation of the loss.

B. SP Module

We propose the Select-Prune Module, which is able to automatically SELECT the best convolution kernel candidates, and PRUNE the least contributional channels in the selected kernel. It works as a substitute for residual block [13] in the original stacked hourglass architecture.

Select Module: As we know, only kernels with "correct" receptive field can respond to the "correct" feature, making the receptive field a critical hyper-parameter to tune. Thanks to dilated convolution, using multiple receptive fields in one convolutional layer can be done without adding too much computation by composing convolutional kernels with different dilation values. However, a straightforward combination of all different dilation values could be wasteful on mobile devices. Instead, an effort to hunt for a most efficient combination in different layers need to be made. The Select Module is designed for such a purpose, to select the convolution kernel with the best receptive fields in a convolutional layer.

The Select Module has several candidate convolution kernels with different dilation values and its task is to select the best one among them. Input features are fed into each kernel and the contribution is evaluated by a MLP block called *Contribution Evaluation Networks (CEN)*, and outcome of all

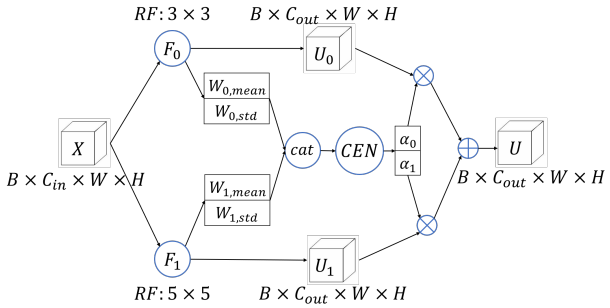


Fig. 4. The structure of Select Module. B: batch size, C_{in} and C_{out}: number of the input/output feature maps' channels, W and H: the size of simple feature map, RF: receptive field.

candidate kernels are weighted by the predicted contribution of the CENs and then summed together as the final outcome of the Select Module. The CENs and the redundant convolution kernel are only involved during the training process, while during inference only the kernel with most average contribution is kept. In other words, when the optimal dilation value of the Select Module is determined, the module will degenerate into an ordinary dilation convolution, and become easy to implement using any frameworks.

Here we give an example of the Select Module with capacity 2 in Fig. 4. For an input feature $X \in \mathbb{R}^{B\times C_{in}\times W\times H}$, we denote the two convolution kernels as $F_0 : X \to U_0 \in \mathbb{R}^{B\times C_{out}\times W\times H}$ and $F_1 : X \to U_1 \in \mathbb{R}^{B\times C_{out}\times W\times H}$, with the dilation value of 1 and 2 respectively, which means they are just like a basic convolutional layer with a kernel size of 3 and 5 in terms of receptive fields. The input of the l^{th} convolutional layer's CEN is the flattened parameter of all candidate kernels, denoted as $W_{i,mean}$ and $W_{i,std}$.

Although in the example there are only two kernel candidates, the Select Module can be easily applied to the case with multiple dilation values. We have:

$$\begin{aligned} Vector[\alpha^l] &= \mathrm{CEN}(Vector[W^l]) \\ &= Softmax(FC(ReLU(FC(Vector[W^l])))) \end{aligned} \tag{1}$$

and

$$U^l_{output} = \sum_i \alpha^l_i U^l_i, \tag{2}$$

in which α^l_i is the summing weights of each kernel's output tensor of the l^{th} layer.

In order to highlight the optimal kernel, we need to obtain sparse weights for different kernels. Considering that $\sum_i^l \alpha^l_i = 1$ because of the softmax, we designed the following regularization term to do that:

$$L^l_{\mathrm{CEN}} = -log(\sum_i \alpha^{l\,2}_i) \tag{3}$$

because

$$\sum_i \alpha^{l\,2}_i \le \sum_i \alpha^{l\,2}_i + \sum_{i,j} \alpha^l_i \alpha^l_j = (\sum_i \alpha^l_i)^2 = 1 \tag{4}$$

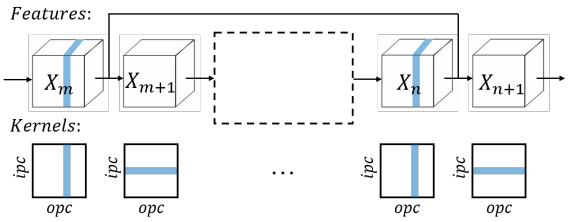


Fig. 5. Pruning in hourglass, blue parts represent pruned channels.

and only when one term α^l_i equals one and others become zero, $\sum_i \alpha^{l\,2}_i$ can reach the maximum value 1 and L^l_{CEN} obtains its minimum value 0. With such regularization, vector $[\alpha^l]$ tends to be sparser as driven by the loss function.

Prune Module: The network is being trained and pruned altenatively [19], splitting the learning procedure into interleaved training phases and pruning phases. The network may go through several epochs during one training phase, and during the following pruning phase, the candidate convolutional channels are traversed, and those that contribute less than a certain level are pruned. The contribution score is calculated by the norm of the weights, as in [20]. In case of that all the channels play reasonable roles and the network stops pruning too early, we decide to constantly prune channels of the "global" minimum contribution among all layers in the network each time, thus the network will continue pruning to achieve every configuration of the trade-offs. To make contributions in different layers comparable, we use softmax to normalize the contribution of different channels in the same layer before compared with channels from other layers.

In short, the contribution of the i^th channel in l^{th} layer is calculated as:

$$contribution^l_i = \frac{exp(\|W^l_i\|_2)}{\sum^l_j exp(\|W^l_j\|_2)} \tag{5}$$

and there are two types of channel that should be pruned:

- The channel's contribution is too small in its layer, e.g., less than 1%;
- The channel's contribution is one of the smallest 5 in the whole model.

For a simple convolutional network, using the above pruning strategy is sufficient. But since the hourglass architecture used in this paper has skip connections, special considerations are needed to keep the manifold consistent. The approach is shown in Fig. 5.

Pruning the n-th convolution output affects not only the input of the $(n+1)$-th layer, but also the input and output channels of the layer from skip connections. Like the method mentioned in [21], the convolution kernels connected by skip connections are grouped into the same group, and the score is the sum of the pruning scores of each convolution kernel in the group.

Similarly, we use the $L1$ norm of each channel's weight to make them sparse in order to prune the unimportant channels.

C. Slots Inference

The network's input is a bird eye view (BEV) image, and the output are estimated slot corners and marking lines (including entry lines and separating lines).

The surround-view image sequence could be synthesized in real-time during inference from outputs of multiple wide-angle cameras mounted on the vehicle. Four fish-eye cameras are used, and the surround-view is the composite view of them: the front, left, rear and right view. A lookup table is used to calculate the relations between the fish-eye image and the BEV. The transformation matrix from the BEV image coordinate system to the world coordinate system originating at the vehicle center is calibrated beforehand. Once a parking slot is detected in the BEV, its world coordinates can be calculated. Further details about BEV image synthesis can be found in [1].

We first detect the corners of slots by finding local maximums on the corner heatmap. Then, corners are paired and screened with several rules to decide whether the two corners belong to one slot. Many prior and posterior knowledge can serve as the rules. For example, entrance corners should be within reasonable distances, and no third corner points should intercept between them; standard parking slots form angles of 90°, 45° or 135° between entrance line and separating lines, and the average scores along proposed separating lines on those direction can be verified for testing the existence of the lines. The filtered lines and corners are finally assembled together as the detected parking slots.

IV. EXPERIMENTS

A. Dataset

We train and test on the challenging DeepPS [1] dataset, which includes various scenarios, weather, time and parking slot types. There are 9527 training images and 2138 validating BEV images, and the dataset can be found at https://cslinzhang.github.io/deepps/.

B. Metrics

Here we follow the definition of Zhang et al. [4]: for a ground-truth marking-point g_i, if there is a detected marking point d_i satisfying $e_i = |g_i - d_i| \leq \delta$, where e_i is the localization error and δ is a predefined threshold, we deem that g_i is correctly detected and d_i is a true positive. In this paper, we evaluate the performance of the model by the *accuracy* of slot corner detection with the tolerance of 6 cm by default (approximately 1.5 pixels in the image), instead of 16 cm which was set by Zhang et al. [1] [4]. 1pixel = 4cm

C. Training Details

The input images for training are 224×224 BEV gray-scale images, with corner points and slot line labels. The whole training process can be divided into three stages, including a pre-training stage, a selecting stage, and a pruning stage. The Adam optimization algorithm is adopted with learning rate set to 1e-3 through all the stages, while different stages leverage different loss items.

The performance of an object detection problem is often affected by extreme class imbalance. For the two-class problem in this paper, specifically, the problem is significant since most pixels are not the desired corners or lines, contributing to most of the loss. Therefore, Focal Loss is used instead of the standard Cross Entropy Loss such that the imbalance problem is solved to some extent:

$$L_{heatmaps} = \begin{cases} -(1-p)^2 log(p), & \text{if around}(y == 1) \\ -(1-y)^4 p^2 log(1-p), & \text{otherwise} \end{cases} \tag{6}$$

where p means the predictive value and y means the ground-truth value. The total loss is listed as below:

$$\begin{aligned} L_{total} = & L_{heatmap}^{corners} + L_{heatmap}^{entry-lines} + \lambda_{sl} L_{heatmap}^{side-lines} \\ & + \lambda_{\text{CEN}} L_{\text{CEN}} + \lambda_{L1} L_1 \end{aligned}, \tag{7}$$

where a suppression factor for losses of the less sensitive separating lines is set to $\lambda_{sl} = 0.1$, the regularization term for Contribution Evaluation Networks and the $L1$ regularization for pruning is set to $\lambda_{L1} = 0.05$.

In the pre-training stage, we use 5 epochs for warm-up with only heat-map losses. The model will continue to train 10 epochs with L_{CEN} to L_{total} added, before entering the selecting stage where we determine dilation values of kernels. The decision is made one by one from the first layer to the last and around 2 to 3 training epochs for each layer. In the pruning stage, we remove the L_{CEN}, add L_1 and train the model for at least another 100 epochs. After each epoch, the global contribution of each channel will be calculated according to formula (5) and we use the pruning strategy described in the III-B to remove redundant channels. We finally fine-tuned the network by removing both two regularizations and train last 15 epochs to eliminate their effects.

D. Structure Comparison

To determine the number of hourglasses to be stacked, we first trained both twice stacked hourglasses with intermediate supervision and single hourglass network. According to the experiment results, although multi-hourglass network can improve a bit accuracy and the convergence (收敛) speed, single-hourglass network is sufficient for the task in this paper. Considering that we want to perform real-time detection on CPU, we finally opt for the single-hourglass network.

E. Selecting Kernels

Since the network needs to be pruned after selecting, the initial number of channels is set to 64, thereby providing a larger search space.

Different convolution kernel sizes collect information in different receptive fields. The same receptive field can be achieved by either using standard convolution or using dilated convolution with smaller parameter size. In practice, the kernel with less parameters and computation may or may not run faster than the component. In this work, when the feature size is less than 28×28, dilated convolution with a fixed kernel size

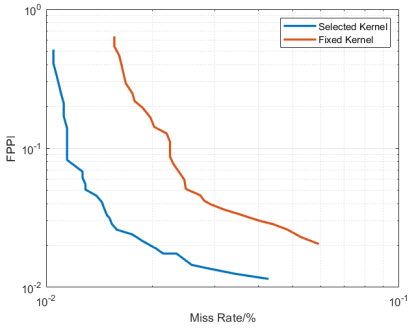


Fig. 6. Comparison of selected and fixed kernel networks

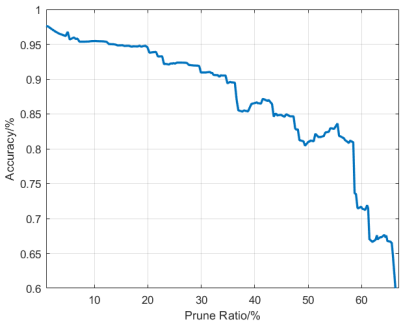


Fig. 7. Location Accuracy of Key Points in the pruning process.

of 3×3 is used for better efficiency, standard convolution is used otherwise because the pytorch implementation (version 1.2.0) of standard convolution is optimized and actually runs faster at larger feature sizes.

We trained two networks under the same conditions, except that one using the selected kernel and the other using fixed 3×3 standard convolution. The comparison of their Miss Rate vs. False Positive Per Image profile is shown in Fig. 6. Obviously, selected receptive fields achieve better performance.

F. Pruning Channels

Fig. 7 shows the model accuracy during the learning (training & pruning) process. Accuracy decreases more and more rapidly when prune ratio increases.

To find a balance between network speed and accuracy, we chose three different pruning configurations for fine-tuning, whose inference time is approximately 30 ms, 22 ms and 17 ms respectively. The accuracy of the model without pruning is added as a baseline (with process time of 52 ms). As is shown in Fig. 8, after 50 epochs of fine-tuning, accuracy of the three pruned networks is generally restored to the level before pruning. It can be noticed that the accuracy of the lighter network goes slightly down in the later training epochs and we have the best parameters at about the 34th epoch.

G. Evaluation and Comparison with Other Methods

We calculated the localization error of the parking slot markings on the validate set and compared it with the results of other object detection methods, The result is shown in Table I.

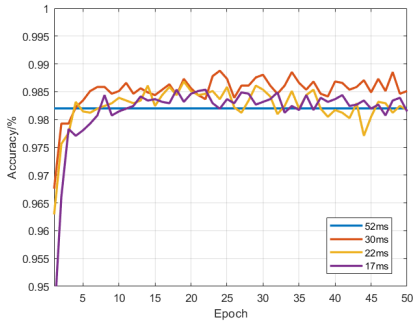


Fig. 8. Accuracy in fine-tuning process.

TABLE I
LOCALIZATION ERROR OF DIFFERENT METHODS (OTHER RESULTS COME FROM [4]).

Method	Localization Error (Unit: cm)
Faster R-CNN [17]	6.12±3.87
SSD [22]	2.52±1.95
DeepPS (YOLO v2-based) [4]	2.58±1.75
Our Method	1.51±2.14

For the overall slot detection, the comparison of the results can be seen in Table II. Compared with the previous methods, our method has gained a lot of reduction in the amount of parameters while sacrificing a bit of accuracy, which makes it easier to arrange into limited storage space and computing power such as on-board equipment.

The frame rate of the model with an input image size of 224×224 can reach 30.9FPS on the CPU to meet real-time detection requirements and it reaches about 150FPS on GPU with the post-processing algorithms are still running on the CPU.

Fig. 9 visualizes four examples of the detection results, in which the corners are marked with blue points. Also the pruned network can recognize parking slots with non-right angled separating lines and entry lines. The model can also identify corners that doesn't belong to any slot, which is shown in the figure but excluded from the slot list during post-processing.

In the above examples, it suggests that by leveraging both point and line detection, the result can be improved under challenging situations. Yet when line detection fails or yields poor results, the parking slot is likely to be excluded from the final result. However, from a practical point of view, here the

TABLE II
ACCURACY OF DIFFERENT METHODS.

Method	Parameter size/MB	Precision	Recall
PSD_L [1] (16cm)	8.38 MB	98.55%	84.64%
DeepPS [4] (16cm)	255 MB	**99.54%**	**98.89%**
Our Method (6cm)	**2.39 MB**	98.01%	97.31%
Our Method (16cm)	**2.39 MB**	98.26%	97.56%

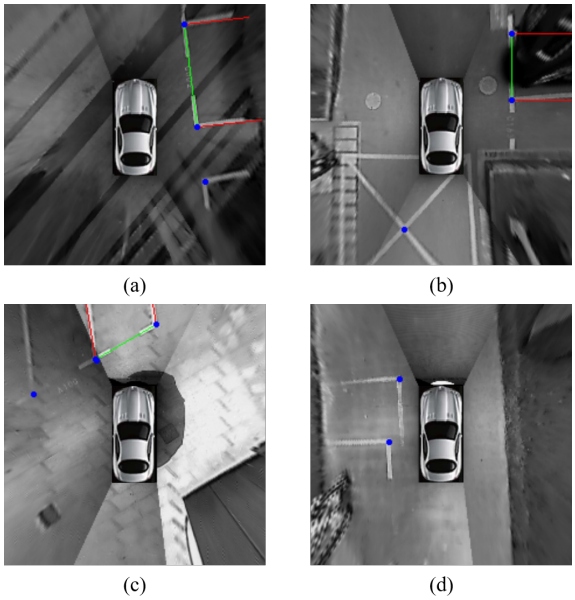


Fig. 9. Different results under shadows and interference: (a) yields correct corners and slots; (b) encounters a misleading ground marking and gives a wrong corner, but excludes it during slot inference; (c) yields the correct corners yet misses one slot as a result of the ground marking degradation; (d) gets completely wrong corners because of the misleading ground marking, yet successfully excludes the corners as no proper slot can be formed.

missed slot detection (false negative) is more acceptable than the false detection (false positive).

V. CONCLUSIONS

In this paper, we propose a deep-learning-based model to detect the parking corners and lines for slots inference. The hourglass architecture is used to generate point and line features, and the features are leveraged jointly for parking slot detection. The result is quite promising in accelerating fully convolutional networks without complicate architecture searching methods.

At the same time, the proposed effective network design solution, using the SP module to automatically outline the shape of the network and choose the optimal convolutional kernel configurations during training, can be borrowed for many other applications. The SP module can directly replace the convolution blocks in any network, and get free performance improvement by sacrificing some training speed but without additional inference overhead.

ACKNOWLEDGMENT

This work is funded by the Open Project of State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body with funding number 31815005 and by the he National Natural Science Foundation of China with funding number 61906138.

REFERENCES

[1] Lin Zhang, Xiyuan Li, Junhao Huang, Ying Shen, and Dongqing Wang. Vision-Based Parking-Slot Detection: A Benchmark and A Learning-Based Approach. *Symmetry*, 10:64, March 2018.

[2] Tom Bruls, Horia Porav, Lars Kunze, and Paul Newman. The right (angled) perspective: Improving the understanding of road scenes using boosted inverse perspective mapping. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, page 302–309, Jun 2019.

[3] Yan Wu, Tao Yang, Junqiao Zhao, Linting Guan, and Wei Jiang. VH-HFCN based Parking Slot and Lane Markings Segmentation on Panoramic Surround View. *arXiv:1804.07027 [cs]*, April 2018. arXiv: 1804.07027.

[4] Lin Zhang, Junhao Huang, Xiyuan Li, and Lu Xiong. Vision-Based Parking-Slot Detection: A DCNN-Based Approach and a Large-Scale Benchmark Dataset. *IEEE Transactions on Image Processing*, 27:5350–5364, November 2018.

[5] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked Hourglass Networks for Human Pose Estimation. *arXiv:1603.06937 [cs]*, March 2016.

[6] Jin Xu, Guang Chen, and Ming Xie. Vision-guided automatic parking for smart car. In *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511)*, pages 725–730, Dearborn, MI, USA, 2000. IEEE.

[7] Chunxiang Wang, Hengrun Zhang, Ming Yang, Xudong Wang, Lei Ye, and Chunzhao Guo. Automatic Parking Based on a Bird's Eye View Vision System. *Advances in Mechanical Engineering*, 6:847406, January 2014.

[8] Jae Kyu Suhr and Ho Gi Jung. Sensor Fusion-Based Vacant Parking Slot Detection and Tracking. *IEEE Transactions on Intelligent Transportation Systems*, 15:21–36, February 2014.

[9] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. *arXiv:1612.08242 [cs]*, December 2016. arXiv: 1612.08242.

[10] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size. *arXiv:1602.07360 [cs]*, February 2016. arXiv: 1602.07360.

[11] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. *arXiv:1905.02244 [cs]*, May 2019. arXiv: 1905.02244.

[12] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. *arXiv:1807.11164 [cs]*, July 2018. arXiv: 1807.11164.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. arXiv: 1512.03385.

[14] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. *arXiv:1608.06993 [cs]*, August 2016. arXiv: 1608.06993.

[15] Robert J. Wang, Xiang Li, and Charles X. Ling. Pelee: A Real-Time Object Detection System on Mobile Devices. *arXiv:1804.06882 [cs]*, April 2018. arXiv: 1804.06882.

[16] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective Kernel Networks. *arXiv:1903.06586 [cs]*, March 2019.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497 [cs]*, June 2015. arXiv: 1506.01497.

[18] Jun-Tae Lee, Han-Ul Kim, Chul Lee, and Chang-Su Kim. Semantic line detection and its applications. *ICCV*, page 9, 2017.

[19] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning Convolutional Neural Networks for Resource Efficient Inference. *arXiv:1611.06440 [cs, stat]*, November 2016. arXiv: 1611.06440.

[20] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning Filters for Efficient ConvNets. *arXiv:1608.08710 [cs]*, August 2016. arXiv: 1608.08710.

[21] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate Decorator: Global Filter Pruning Method for Accelerating Deep Convolutional Neural Networks. *arXiv:1909.08174 [cs, eess]*, September 2019. arXiv: 1909.08174.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. *arXiv:1512.02325 [cs]*, 9905:21–37, 2016. arXiv: 1512.02325.